

Matt's Code

Matt James

Contents

1	Introduction	1
1.1	Setting up the environment	1
1.1.1	Linux	1
1.1.2	Windows	2
1.1.3	MacOS	2
1.2	Setting up a virtual environment	2
1.3	Some python packages	2
2	Plasma Models	5
2.1	spicedmodel: The Scalable Plasma Ion Composition and Elec- tron Density Model	5
2.2	HermeanFLRModel: Model of Mercury’s dayside plasma mass den- sity	5
2.3	PyGCPM: Wrapper for the Global Core Plasma Model	5
3	Field Models	7
3.1	PyGeopack: Python wrapper for the Tsyganenko field models . .	8
3.2	geopack: C++ wrapper for Tsyganenko field models	8
3.3	libinternalfield: C++ spherical harmonic model code	8
3.4	vsmodel: Python based Volland-Stern electric field model for Earth	8
3.5	JupiterMag: Python wrapper for Jovian field models	8
3.6	libjupitermag: C++ library for field tracing in Jupiter’s mag- netosphere	8
3.7	con2020: Python implementation of Jupiter’s magnetodisc model	8
3.8	libcon2020: C++ implementation of Jupiter’s magnetodisc model	8
3.9	jrm33: The JRM33 model in Python	8
3.10	jrm09: The JRM09 model in Python	8
3.11	vip4model: The VIP4 model in Python	8
4	Spacecraft Data	9
4.1	Arase: Download and read Arase data	9
4.2	RBSP: Download and read Van Allen Probe data	9
4.3	cluster: Download and read Cluster data	9
4.4	pyCRRES: Download and read CRRES data	9
4.5	themissc: Download and read THEMIS data	9
4.6	imageeuv: Download and read IMAGE EUV data	9
4.7	imagerpi: Download and read IMAGE RPI data	9
4.8	imagePP: Download and read Goldstein’s plasmopause dataset .	9

4.9	PyMess: Download and read MESSENGER data	9
4.10	FIPSProtonData: Download and read ANN verified FIPS moments	9
4.11	VenusExpress: Download and read VEX data	9
5	Ground Data and Geomagnetic Indices	11
5.1	groundmag: Tools for processing and reading ground magnetometer data	11
5.2	SuperDARN: Simple SuperDARN fitacf reading code	11
5.3	kpindex: Download the latex Kp indices	11
5.4	pyomnidata: Download the latex OMNI and solar flux data	11
5.5	smindex: Read the SuperMAG indices	11
6	Machine Learning	13
6.1	NNClass: Simple neural network classifier module	13
6.2	NNFunction: Train neural networks on arbitrary functions	13
7	Other Tools	15
7.1	wavespec: Spectral analysis tools	16
7.2	MHDWaveHarmonics: Tools for MHD waves	16
7.3	FieldTracing: Python field tracing code	16
7.4	DateTimeTools: Tools for dealing with dates and times	16
7.5	datetime: C++ library dealing for dates and times	16
7.6	PyFileIO: Tools for reading and writing files	16
7.7	RecarrayTools: Tools for manipulating <code>numpy.recarrays</code>	16
7.8	PBSJobExamples: Examples for submitting jobs to PBS	16
7.9	PlanetSpice: SPICE related code	16
7.10	ColorString: Change colour of strings in the terminal	16
7.11	cppembedbinary: Examples for embedding data into C++ code	16
7.12	libspline: C++ library for splines	16
7.13	linterp: C++ interpolation code	16

Chapter 1

Introduction

This document lists a bunch of the GitHub repositories created by me which may be useful to others. Some of these repositories are fairly complete, others are less so. I will do my best to fix and update anything that is buggy or incomplete, please do report bugs in the relevant repositories if you can. If you're feeling particularly helpful - feel free to send pull requests.

Most of the code here is written in Python, some things make use of C++ libraries to do some of the heavy lifting, one is a pretty dodgy Python wrapper of a C++ wrapper of Fortran code... Some of the modules and libraries used here are dependencies of others. In the more complete repos `pip` will take care of dependencies, otherwise some manual installation may be required.

1.1 Setting up the environment

In this section I describe how to set up the environment such that everything *should* pretty much work...

1.1.1 Linux

If running on ALICE/SPECTRE, you will most likely be required to enable the following modules:

```
module load gcc/9.3
module load python/gcc/3.9.10
module load git/2.35.2
```

where exact version numbers may change (use whatever is latest, don't just copy and paste!) and the replacement for SPECTRE/ALICE may have another method for loading these things in for all I know. I also recommend adding those to the end of your `~/.bashrc` file so that they load every login, e.g.:

```
echo module load gcc/9.3 >> ~/.bashrc
echo module load python/gcc/3.9.10 >> ~/.bashrc
echo module load git/2.35.2 >> ~/.bashrc
```

The above is unlikely to be necessary on a local Linux installation, instead I would recommend installing `git`, `gcc`, `g++`, `make`, `gfortran` and `pip3`, e.g. in Ubuntu:

```
sudo apt install git gcc g++ binutils gfortran python3-pip
```

All of the above should allow you to install/run/compile most of my code. I wouldn't recommend using Conda in Linux - I know it has cause some problems/confusion when it comes to linking Python with C/C++ on SPECTRE.

1.1.2 Windows

A fair portion of the code is able to run on Windows - much of the Python code is platform independent and some of the C++ libraries/backends are able to be compiled using Windows. In this case, I *would* actually recommend installing Conda, as it worked for me. The GCC compilers (for C/C++/Fortran) can all be installed easily with TDM-GCC ([get the 64-bit version here](#)), just remember to put a tick in the box for "fortran" and "openmp".

1.1.3 MacOS

I managed to install the relevant packages in a virtual Hackintosh once. I don't remember how, perhaps using homebrew. Good luck...

1.2 Setting up a virtual environment

In SPECTRE I never actually bothered with a virtual environment, mistakes were made, headaches may have been avoided had I done so. This step is entirely optional, but somewhat recommended:

```
#create a virtual environment, call it what you want,  
#here I call mine "env"  
python3 -m venv env
```

Once this has been created, you **MUST** activate it before running any code, or you will just be running things globally:

```
source env/bin/activate
```

note that I am assuming that `env` exists in the current working directory, if not adjust the path accordingly! If it works, the prompt terminal prmpt should change, e.g:

```
#before:  
matt@matt-MS-7B86:~$  
  
source env/bin/activate  
#after:  
(env) matt@matt-MS-7B86:~$
```

1.3 Some python packages

Here are a list of Python packages which are either going to be required by most of my code, or would just be recommended:

1. `ipython` : best Python interpreter, forget notebooks
2. `numpy` : essential, don't skip
3. `matplotlib` : for plotting
4. `scipy` : loads of good stuff here
5. `wheel` : used to build Python packages to be installed by `pip`
6. `cdflib` : reads CDF files
7. `keras` : nice for machine learning
8. `tensorflow` : also machine learning

install them:

```
#update pip first
```

```
python3 -m install pip --upgrade --user
```

```
pip3 install ipython numpy matplotlib scipy wheel cdflib keras tensorflow --user
```

where the “`--user`” flag may or may not be necessary, depending on your version of Python - it places the installed modules in `~/.local/lib/python3.9/site-packages`.

In theory, at this point you should be able to run `ipython3` (or just `ipython`) within the terminal, from which any installed code can be imported. The reason I recommend using `Ipython` over the standard Python interpreter is that it has autocomplete and it uses pretty colours for syntax highlighting. It would also be a good idea to enable the autoreload feature in `Ipython`, which recompiles anything that has been edited since it was last run, otherwise would have to reload the code manually (or restart the session) after every edit. Run

```
ipython profile create
```

then add the following lines to `~/.ipython/profile.default/ipython_config.py`:

```
c.InteractiveShellApp.extensions = ['autoreload']  
c.InteractiveShellApp.exec_lines = ['%autoreload 2']  
c.InteractiveShellApp.exec_lines.append('print("Warning: disable autoreload in ipython_config.py")')
```

That should just about do it.

Chapter 2

Plasma Models

- 2.1 `spicedmodel`: The Scalable Plasma Ion Composition and Electron Density Model
- 2.2 `HermeanFLRModel`: Model of Mercury's day-side plasma mass density
- 2.3 `PyGCPM`: Wrapper for the Global Core Plasma Model

Chapter 3

Field Models

- 3.1 PyGeopack: Python wrapper for the Tsyganenko field models
- 3.2 geopack: C++ wrapper for Tsyganenko field models
- 3.3 libinternalfield: C++ spherical harmonic model code
- 3.4 vsmodel: Python based Volland-Stern electric field model for Earth
- 3.5 JupiterMag: Python wrapper for Jovian field models
- 3.6 libjupitermag: C++ library for field tracing in Jupiter's magnetosphere
- 3.7 con2020: Python implementation of Jupiter's magnetodisc model
- 3.8 libcon2020: C++ implementation of Jupiter's magnetodisc model
- 3.9 jrm33: The JRM33 model in Python
- 3.10 jrm09: The JRM09 model in Python
- 3.11 vip4model: The VIP4 model in Python

Chapter 4

Spacecraft Data

- 4.1 Arase: Download and read Arase data
- 4.2 RBSP: Download and read Van Allen Probe data
- 4.3 cluster: Download and read Cluster data
- 4.4 pyCRRES: Download and read CRRES data
- 4.5 themissc: Download and read THEMIS data
- 4.6 imageeuv: Download and read IMAGE EUV data
- 4.7 imagerpi: Download and read IMAGE RPI data
- 4.8 imagePP: Download and read Goldstein's plasma-pause dataset
- 4.9 PyMess: Download and read MESSENGER data
- 4.10 FIPSProtonData: Download and read ANN verified FIPS moments
- 4.11 VenusExpress: Download and read VEX data

Chapter 5

Ground Data and Geomagnetic Indices

- 5.1 `groundmag`: Tools for processing and reading ground magnetometer data
- 5.2 `SuperDARN`: Simple `SuperDARN` `fitacf` reading code
- 5.3 `kpindex`: Download the latex `Kp` indices
- 5.4 `pyomnidata`: Download the latex OMNI and solar flux data
- 5.5 `smindex`: Read the `SuperMAG` indices

Chapter 6

Machine Learning

- 6.1 NNClass: Simple neural network classifier module
- 6.2 NNFunction: Train neural networks on arbitrary functions

Chapter 7

Other Tools

- 7.1 `wavespec`: Spectral analysis tools
- 7.2 `MHDWaveHarmonics`: Tools for MHD waves
- 7.3 `FieldTracing`: Python field tracing code
- 7.4 `DateTimeTools`: Tools for dealing with dates and times
- 7.5 `datetime`: C++ library dealing for dates and times
- 7.6 `PyFileIO`: Tools for reading and writing files
- 7.7 `RecarrayTools`: Tools for manipulating `numpy.recarrays`
- 7.8 `PBSJobExamples`: Examples for submitting jobs to PBS
- 7.9 `PlanetSpice`: SPICE related code
- 7.10 `ColorString`: Change colour of strings in the terminal
- 7.11 `cppembedbinary`: Examples for embedding data into C++ code
- 7.12 `libspline`: C++ library for splines
- 7.13 `linterp`: C++ interpolation code